Politechnika Wrocławska

Sztuczna inteligencja i inżynieria wiedzy

Laboratorium - Lista 4

13.06.2024

Krzysztof Głowacz, 266545

Spis treści

1 Wstęp

Laboratorium nr 4 stanowiło wstęp do dziedziny uczenia maszynowego (*Machine Learning*). Miało na celu symulację problemu identyfikacji zapotrzebowania na produkty o danych parametrach (na podstawie dostępnych danych z przeszłości). Rozwiązanie takiego problemu wymagało przejścia przez kolejne fazy: eksploracji danych, przygotowania danych, klasyfikacji oraz oceny dokonanej klasyfikacji. Każda ze wspomnianych faz opisana została w kolejnych sekcjach sprawozdania.

2 Eksploracja danych

Zbiór danych zawarty był w pliku **t-shirts.csv** zawierającym 20 tysięcy rekordów, z których każdy miał informacje o: rozmiarze koszulki, jej materiale, kolorze, długości rękawów i samym zapotrzebowaniu. Początkowe 10 rekordów pliku wyglądało następująco:

size	material	color	sleeves	demand
S	nylon	white	long	medium
XL	polyester	cream	short	high
S	silk	blue	short	medium
M	cotton	black	short	medium
XL	polyester	orange	long	medium
XS	polyester	black	short	medium
XXL	polyester	green	short	medium
L	linen	yellow	short	high
XL	linen	cream	long	high
XS	nylon	red	short	high

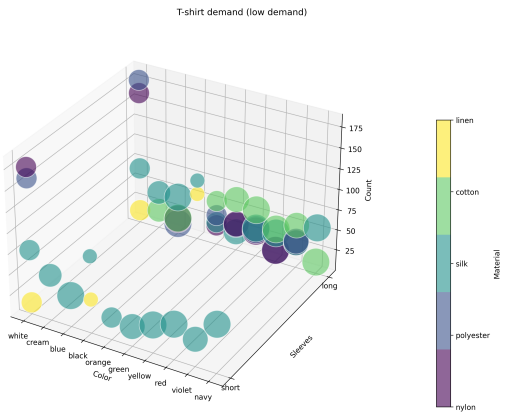
Tabela 1: Przykładowe dane z pliku t-shirts.csv

Eksploracja danych rozpoczęła się od utworzenia wykresów 5D, które miały na celu zobrazować występujące trendy wśród danych. Na zdjęciu nr 1 przedstawione zostały 3 wykresy, osobno dla niskiego, średniego i wysokiego zapotrzebowania na koszulki, na których:

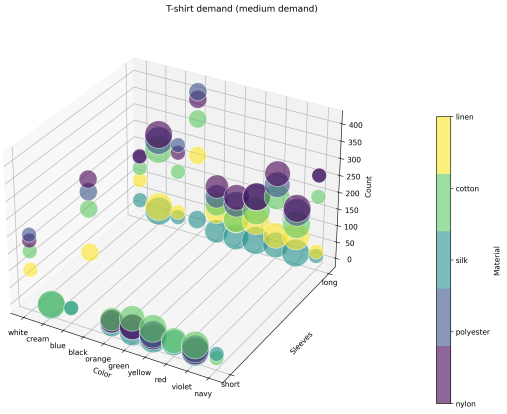
- oś X odpowiada za kolor koszulki
- oś Y odpowiada za długość rękawów
- oś Z odpowiada za liczbę koszulek o danych parametrach w pliku (częstość występowania takiej konfiguracji)
- kolor znacznika odpowiada za rodzaj materiału koszulki (legenda z prawej strony)
- rozmiar znacznika odpowiada za rozmiar koszulki (im większy, tym większy rozmiar danej koszulki)

Na podstawie tak stworzonych wykresów możemy zauważyć pewne trendy. Wśród koszulek o niskim zapotrzebowaniu dominują te wykonane z jedwabiu i bawełny, a także koszulki o największych rozmiarach (*XXL* czy *3XL*). Najwięcej koszulek w tej kategorii, tzn. największą grupą „nietrafionych" produkcji są koszulki w standardowych, średnich rozmiarach wykonane z nylonu i poliestru. Widzimy tutaj także sporą kumulację koszulek, które mają długie rękawy. Koszulki z krótkim rękawem na tym zestawieniu to niemalże wyłącznie koszulki jedwabne, często w większych rozmiarach. W kategorii koszulek o średnim zapotrzebowaniu możemy zaobserwować, że generalnie koszulek z krótkim rękawem także jest niewiele, a jeśli już jakieś się pojawiają, to głównie mają one większe rozmiary. Tylko białych i czarnych koszulek z krótkim rękawem o mniejszych/standardowych rozmiarach jest w tej grupie więcej (występują tu koszulki z różnych materiałów, tzn. lniane, bawełniane, nylonowe i poliestrowe). Wśród koszulek z długimi rękawami dane nie rozkładają się w tak zbliżony do siebie sposób, choć można zauważyć, że najmniejsze liczebności cechują w tej grupie koszulki jedwabne, trochę większe lniane, dalej bawełniane, poliestrowe i nylonowe. Wciąż przeważającą część stanowią koszulki o większych i największych rozmiarach. Kategoria koszulek o wysokim zapotrzebowaniu charakteryzuje się przede wszystkim tym, że nie zawiera w zasadzie żadnych koszulek o dużych rozmiarach. Występują w niej koszulki głównie w rozmiarach średnich i standardowych. Nie ma wśród nich także produktów jedwabnych. Koszulek z krótkim rękawem jest w tej kategorii więcej (odwrotnie do poprzednich). Jeśli spojrzeć na to zestawienie pod względem kolorów

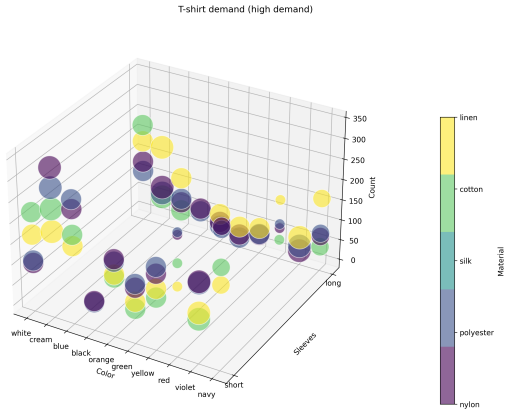
to zauważyć można, że dla koszulek z krótkim rękawem dla większości kolorów najwięcej koszulek jest wykonanych z poliestru/nylonu podczas, gdy dla koszulek z długim rękawem jest to głównie len.



(a) Niskie zapotrzebowanie



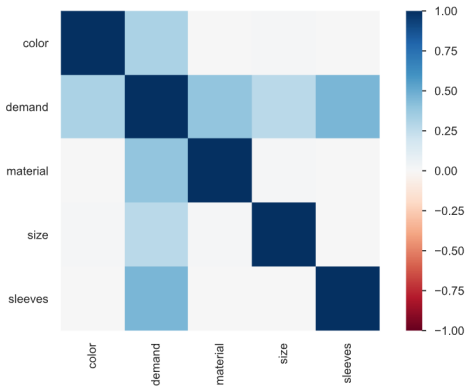
(b) Średnie zapotrzebowanie



(c) Wysokie zapotrzebowanie

Zdjęcie 1: Liczba koszulek o danych parametrach według zapotrzebowania

Innym rodzajem wstępnej analizy danych jest skorzystanie z biblioteki `ydata_profiling`, która umożliwia automatyczne stworzenie czytelnego i przejrzystego raportu, który w łatwy sposób można wyeksportować, np. do formatu *html*. Tabela 2 przedstawia dane z raportu dotyczące częstości występowania każdej z wartości z danej kategorii w źródłowym pliku. Raport ten wygenerował także macierz korelacji cech (*heatmap*):



Zdjęcie 2: Macierz korelacji poszczególnych atrybutów koszulek

Wartość	Liczba	Częst.
L	4408	22.0%
XL	3480	17.4%
M	3414	17.1%
XXL	2614	13.1%
S	2585	12.9%
XS	1764	8.8%
3XL	1735	8.7%

(a) Rozkład rozmiarów koszulek

Wartość	Liczba	Częst.
nylon	5652	28.3%
polyester	5555	27.8%
cotton	4334	21.7%
linen	3336	16.7%
silk	1123	5.6%

(b) Rozkład materiałów koszulek

Wartość	Liczba	Częst.
white	3286	16.4%
black	3118	15.6%
cream	2298	11.5%
navy	2289	11.4%
red	2000	10.0%
blue	1958	9.8%
yellow	1401	7.0%
orange	1320	6.6%
violet	1295	6.5%
green	1035	5.2%

(c) Rozkład kolorów koszulek

Wartość	Liczba	Częst.
long	10117	50.6%
short	9883	49.4%

(d) Rozkład rękawów koszulek

Wartość	Liczba	Częst.
high	8965	44.8%
medium	8709	43.5%
low	2326	11.6%

(e) Rozkład zapotrzebowania na koszulki

Tabela 2: Rozkłady częstości występowania wartości dla każdego atrybutu koszulek

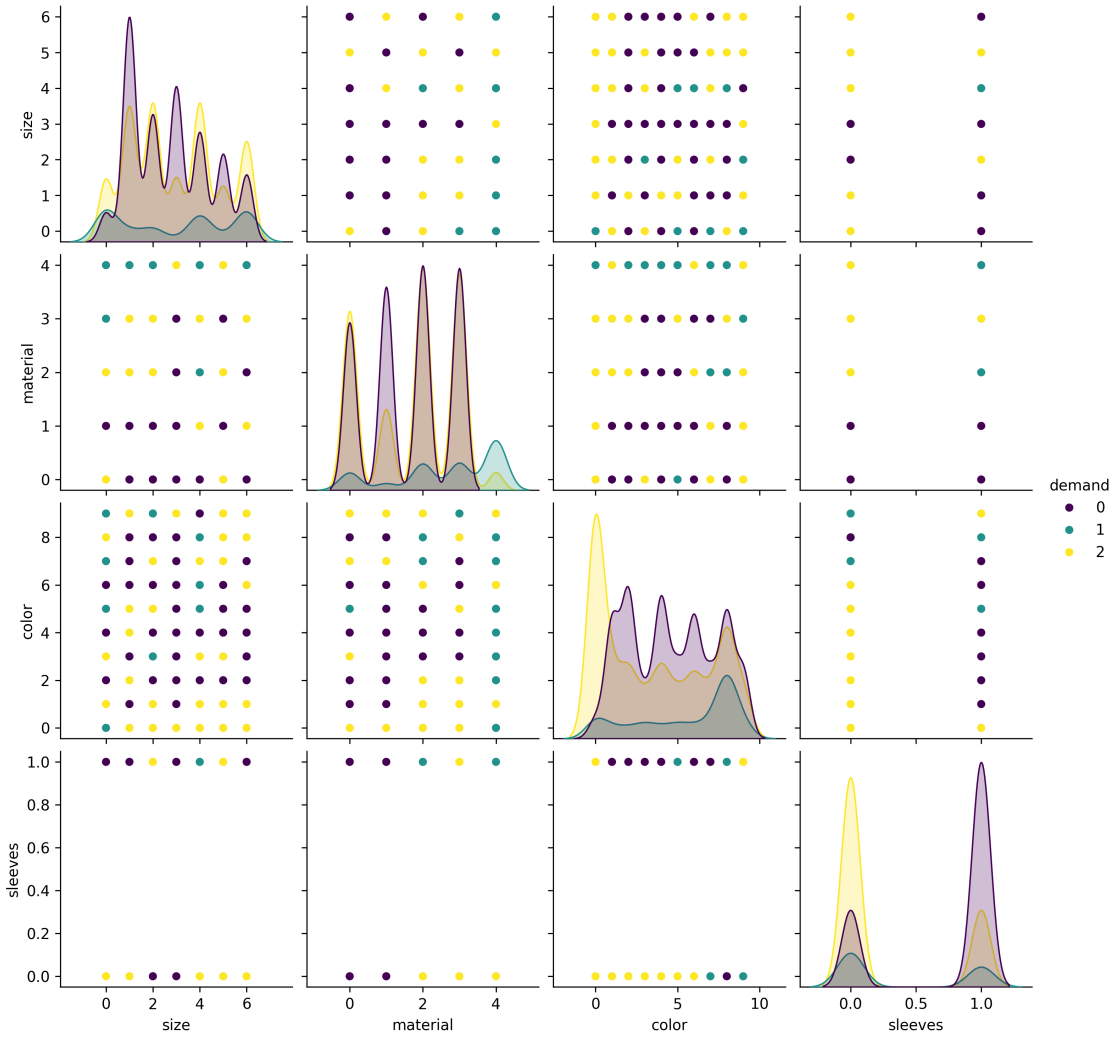
Ostatnią z wykorzystanych metod wstępnej analizy danych jest wykres stworzony przy użyciu biblioteki seaborn, który przedstawia rozkład zapotrzebowania na koszulki w zależności od każdej z par atrybutów. W celu jego wykonania, a także wykonania kolejnych etapów laboratorium, konieczne było zmapowanie słownych wartości cech koszulek na liczby całkowite, co odbyło się według poniższego schematu:

```
Size column mapping: {0: '3XL', 1: 'L', 2: 'M', 3: 'S', 4: 'XL', 5: 'XS', 6: 'XXL'}

Material column mapping: {0: 'cotton', 1: 'linen', 2: 'nylon', 3: 'polyester', 4: 'silk'}

Color column mapping: {0: 'black', 1: 'blue', 2: 'cream', 3: 'green', 4: 'navy', 5: 'orange',
    6: 'red', 7: 'violet', 8: 'white', 9: 'yellow'}

Sleeves column mapping: {0: 'long', 1: 'short'}

Demand column mapping: {0: 'high', 1: 'low', 2: 'medium'}
```

Rezultat (wykres) przedstawiony został na zdjęciu 3.

Na podstawie zawartej wstępnej analizy danych można wyciągnąć następujące wnioski:

- najsilniejszą korelacją z zapotrzebowaniem wyróżnia się rodzaj rękawów koszulki, najsłabszą natomiast jej rozmiar [Zdjęcie 2],
- wartości w obrębie prawie każdej z cech koszulek są nierównomiernie rozłożone, wyjątkiem jest rodzaj rękawów koszulki. W niektórych cechach wartości występujące najczęściej stanowią nawet pięć razy liczniejszą grupę niż te, które występują najrzadziej [Tabela 2],
- zgodnie z przypuszczeniami największe zapotrzebowanie dotyczy koszulek w standardowych, średnich rozmiarach, przede wszystkim tych z krótkim rękawem [Zdjęcie 3].



Zdjęcie 3: Zapotrzebowanie na koszulki o danej kombinacji atrybutów dla każdej z możliwych par

3 Przygotowanie danych

W celu porównania wpływu przygotowania danych na późniejszą klasyfikację dane zostały podzielone na zestawy - uczący i walidacyjny (w proporcji 8:2) przy pomocy pięciu różnych metod:

```
preprocessing_methods = {
    "none": lambda X: X,
    "normalization": MinMaxScaler(),
    "standardization": StandardScaler(),
    "feature_selection": SelectKBest(score_func=f_classif, k=2),
    "pca": PCA(n_components=2)
}
```

Pierwsza z nich oznaczała brak jakiegokolwiek przekształcenia, a więc te dane w „surowej" formie trafią później do klasyfikatora. Druga metoda (normalizacja) ma na celu znormalizować dane, tj. sprowadzić wszystkie wartości do przedziału $[0, 1]$. Trzecia (standaryzacja) polega na przekształceniu danych wejściowych do rozkładu o średniej $\mu = 0$ i odchyleniu standardowym $\sigma = 1$. Zarówno normalizacja, jak i standaryzacja mają na celu zapobiegnięciu sytuacji, w której trenowanie danych będzie w pewien sposób stronnicze (*ang. biased*) ze względu na potencjalnie nieprzeskalowane wartości, które mogą przyczynić się do słabszego dopasowania modelu na etapie klasyfikacji. Czwarta metoda (selekcja cech) polega na wybraniu k najlepszych cech, wybranych na podstawie wartości funkcji oceniającej (tu $k = 2$, a funkcją

oceniającą jest test statystyczny polegający na analizie wariancji *ANOVA F-test*). Ostatnia z metod (analiza głównych składowych) jest najbardziej zaawansowana i dotyczy redukcji wymiarowości danych (wybierane są tzw. główne składowe). Metoda ta może być użyteczna do wizualizacji, usuwania szumu, czy samego przyspieszania kosztownych algorytmów uczenia maszynowego.

4 Klasyfikacja

Kolejną częścią laboratorium była właściwa klasyfikacja danych. Aby ją przeprowadzić konieczne było przygotowanie klasyfikatorów. W tym przypadku zdecydowano się na trzy różne klasyfikatory:

1. **naiwny klasyfikator Bayesa (*naive Bayes classifier*)**
   Probabilistyczny klasyfikator oparty o twierdzenie Bayesa, który naiwnie zakłada niezależność cech wśród danych. Polega na obliczaniu dla każdej klasy prawdopodobieństwa zdarzenia, że dana próbka należy do danej klasy na podstawie wcześniej obliczonych prawdopodobieństw warunkowych. Jest prosty i łatwy w użyciu, może sobie dobrze radzić z wielowymiarowymi danymi. Może mieć natomiast problem w przypadku silnie skorelowanych danych. Dodatkowo rzadko kiedy założenie o niezależności cech jest w praktyce spełnione.

2. **drzewo decyzyjne (*decision tree*)**
   Algorytm klasyfikacji wykorzystujący strukturę drzewa binarnego do podejmowania kolejnych decyzji na podstawie wartości cech. Dane na każdym poziomie dzielone są coraz bardziej na mniejsze podzbiory na podstawie wartości cech, zaczynając od cech najbardziej dzielących i różnicujących dane. Ten rodzaj klasyfikatora jest dosyć intuicyjny, a także szybki w użyciu. Dobrze radzi sobie z danymi ciągłymi, a także skategoryzowanymi. Jego wadą jest jednak podatność na przetrenowanie (*overfitting*), a także potencjalna kosztowność tworzenia rozbudowanego drzewa.

3. **klasyfikator wektorów nośnych (*Support Vector Machine*)**
   Algorytm klasyfikacji, którego celem jest znalezienie hiperpłaszczyzny najlepiej oddzielające klasy w przestrzeni cech, tj. hiperpłaszczyzny maksymalizującej margines między klasami, gdzie margines oznacza najmniejszą odległość między dwiema próbkami z dwóch klas (próbka na granicy marginesu nazywana jest wektorem nośnym). Często w jego przypadku wykorzystuje się jądro (*kernel*), aby dokonać przekształcenia danych do wyższego wymiaru w celu lepszego ich rozdzielenia. Ten klasyfikator może być skuteczny w przypadku danych wielowymiarowych, a także w sytuacji, gdy granice decyzyjne są nieliniowe. Trudnością natomiast będą w tej sytuacji: wybór jądra i parametrów, koszt obliczeniowy, wrażliwość na brak przeskalowania danych.

W celu przetestowania różnych zestawów hiperparametrów skorzystano z narzędzia do ich optymalizacji - `GridSearchCV`. Opiera się on na stworzeniu siatki możliwych kombinacji, która następnie jest przeszukiwana, oceniana (na podstawie zdefiniowanej funkcji oceny), aby na koniec możliwe było podjęcie decyzji i wybranie najlepszego możliwego zestawu hiperparametrów dla wybranego klasyfikatora i analizowanych danych. Zdefiniowane więc zostały różne wartości poszczególnych parametrów dla każdego z trzech klasyfikatorów:

```
param_grids = {
    'naive_bayes': {
        'fit_prior': [True, False],
        'class_prior': [None, [0.3, 0.3, 0.4]],
        'alpha': [0.01, 0.1, 1.0]
    },
    'decision_tree': {
        'criterion': ['gini', 'entropy'],
        'max_depth': [None, 10, 15],
        'min_samples_split': [2, 5, 10]
    },
    'svm': {
        'kernel': ['rbf', 'poly']
    }
}
```

W przypadku naiwnego klasyfikatora Bayesa parametry te dotyczą:

- `fit_prior` - określa, czy algorytm sam ma wyliczyć wagi/prawdopodobieństwa klas, czy też nie (wówczas rozkład jednostajny jest używany),
- `class_prior` - gdy jest określony, to algorytm nie oblicza wag/prawdopodobieństwa klas, ale trzyma się wartości podanych przez użytkownika,
- `alpha` - kontroluje poziom wygładzenia Laplace'a (*Laplace smoothing*), co może mieć wpływ w przypadku występowania zer (danych z zerowym prawdopodobieństwem).

Dla drzewa decyzyjnego parametry te dotyczą:

- `criterion` - definiuje funkcję używaną do oceny jakości podziału drzewa (*Gini impurity* lub entropia - przyrost informacji),
- `max_depth` - określa maksymalną możliwą głębokość drzewa, jeśli nie jest sprecyzowana, to algorytm zakończy się w momencie, gdy wszystkie liście są „czyste" (odseparowane elementy wyłącznie z jednej klasy) lub gdy wszystkie liście zawierają mniej próbek niż wskazuje na to wartość `min_samples_split`,
- `min_samples_split` - minimalna liczba próbek wymagana do dokonania podziału węzła drzewa.

Hiperparametry dla maszyny wektorów nośnych to:

- `kernel` - definiuje jądro użyte podczas zmiany wymiarowości danych.

Dla ostatniego klasyfikatora ubogość kombinacji wynika z jego obliczeniowej kosztowności, która sprawia, że dla większej liczby możliwych wartości parametrów wykonanie eksperymentów trwałoby niezwykle długo.
Siatka poszukiwania optymalnego zestawu hiperparametrów została utworzona w następujący sposób:

```
grid_searches = {
    name: GridSearchCV(estimator=classifier, param_grid=param_grids[name], cv=5, verbose=1,
        scoring=make_scorer(f1_score, average='macro'))
    for name, classifier in classifiers.items()
}
```

Parametr `cv=5` oznacza, że zostanie użyta 5-krotna walidacja krzyżowa, która ma na celu zmniejszenie ryzyka przeuczenia modelu. Parametr `scoring` natomiast definiuje funkcję, jaka zostanie użyta do oceny jakości danego zestawu hiperparametrów. W tym przypadku będzie nią metryka *f1-score*, a więc średnia harmoniczna pomiędzy precyzją i czułością. Zdecydowano się na tę metrykę, ponieważ może być ona dobrym balansem w dziedzinie planowania zapotrzebowania na koszulki, gdzie wysoki jest koszt przewidywań fałszywie pozytywnych (zła identyfikacja trendów), a także koszt przewidywań fałszywie negatywnych (brak identyfikacji danego trendu).

Po uruchomieniu przeszukiwania można sprawdzić, że algorytm faktycznie testuje po kolei różne kombinacje hiperparametrów zanim wybierze optymalny ich zestaw, co widoczne jest na zdjęciu 4.

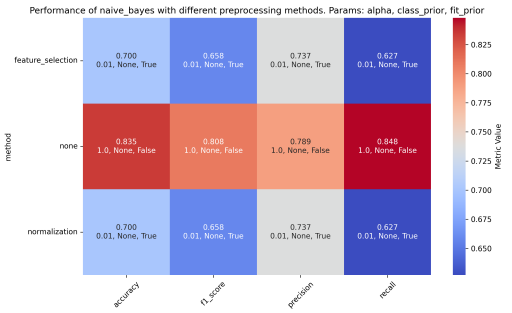
Łącznie algorytm wykonywał się ponad 7 minut ze względu na to, że dla samych drzew decyzyjnych należało sprawdzić 18 różnych kombinacji, a przy 5-krotnej walidacji dawało to 90 różnych dopasowań, które trzeba było przeprowadzić dla 5 różnych sposobów przygotowania danych, a więc tylko dla drzewa decyzyjnego konieczne było wykonanie 450 ewaluacji. Mimo tego i tak to maszyna wektorów nośnych odpowiadała za znaczącą większość czasu obliczeń (choć do sprawdzenia były dla niej zaledwie dwie kombinacje hiperparametrów).

	method	classifier	params	mean_score
0	none	naive_bayes	{'alpha': 0.01, 'class_prior': None, 'fit_prio...	0.741828
1	none	naive_bayes	{'alpha': 0.01, 'class_prior': None, 'fit_prio...	0.784772
2	none	naive_bayes	{'alpha': 0.01, 'class_prior': [0.3, 0.3, 0.4]...	0.769965
3	none	naive_bayes	{'alpha': 0.01, 'class_prior': [0.3, 0.3, 0.4]...	0.769965
4	none	naive_bayes	{'alpha': 0.1, 'class_prior': None, 'fit_prior...	0.741828
5	none	naive_bayes	{'alpha': 0.1, 'class_prior': None, 'fit_prior...	0.784772
6	none	naive_bayes	{'alpha': 0.1, 'class_prior': [0.3, 0.3, 0.4],...	0.769965
7	none	naive_bayes	{'alpha': 0.1, 'class_prior': [0.3, 0.3, 0.4],...	0.769965
8	none	naive_bayes	{'alpha': 1.0, 'class_prior': None, 'fit_prior...	0.741734
9	none	naive_bayes	{'alpha': 1.0, 'class_prior': None, 'fit_prior...	0.785254
10	none	naive_bayes	{'alpha': 1.0, 'class_prior': [0.3, 0.3, 0.4],...	0.769879

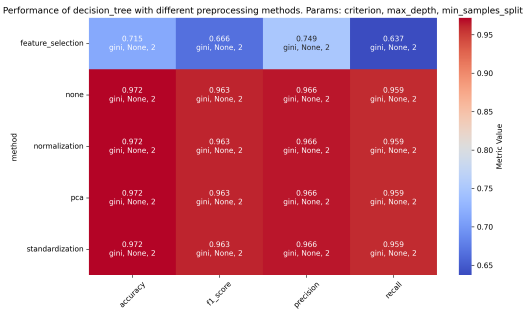
Zdjęcie 4: Przykład poszukiwania optymalnych parametrów dla naiwnego klasyfikatora Bayesa przy braku przygotowania danych

5 Ocena klasyfikacji

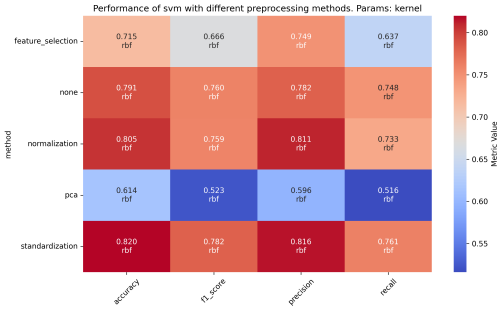
Do porównania wyników i oceny klasyfikacji wykorzystane zostały metryki opisane w instrukcji listy laboratoryjnej. W celu zwiększenia czytelności wyniki przedstawione zostały w postaci trzech macierzy - każda dla innego klasyfikatora. Na jednej z osi macierzy znajdują się użyte metody przygotowania danych, a na drugiej cztery różne metryki. Wartościami na przecięciach metody przygotowania danych i metryki są wyniki - wartości metryk - określające dokładność predykcji wytrenowanego modelu (dla danych przekształconych daną metodą) w stosunku do rzeczywistości, przy czym dokładność rozumiana jest tutaj jako liczba opisująca jakość predykcji w kontekście danej metryki.



(a) Naiwny klasyfikator Bayesa



(b) Drzewo decyzyjne



(c) Maszyna wektorów nośnych

Zdjęcie 5: Wyniki - wartości metryk (w zależności od użytej metody przygotowania danych) dla każdego z klasyfikatorów

Na podstawie powyższych wykresów możemy stwierdzić, że najlepiej z problemem klasyfikacji poradziło sobie drzewo decyzyjne, które nie miało ograniczeń nałożonych na maksymalną głębokość, liczba minimalnych próbek potrzebnych do wykonania podziału węzła drzewa równa była 2 (minimalna możliwa), a funkcją oceny jakości podziału było *Gini impurity*. Dla wszystkich metryk i prawie wszystkich metod przygotowania danych, poza selekcją cech, wartości metryk przekraczały 0.95, co jest bardzo dobrym wynikiem. Powtarzalność wskaźników sugeruje, że drzewo decyzyjne wyczerpując wszystkie możliwości jest bardziej odporne na zmiany we wstępnym przetworzeniu danych. Znacząco niższy wynik dla selekcji cech może z kolei oznaczać, że usunięcie pewnych cech negatywnie wpłynęło na wytrenowanie modelu.

Wyniki osiągane przez maszynę wektorów nośnych plasują ją na drugim miejscu. Najlepsze wyniki zostały osiągnięte dla danych poddanych standaryzacji, niewiele gorsze dla tych poddanych normalizacji. Najgorzej klasyfikator ten poradził sobie przy selekcji cech i analizie głównych składowych - zastosowane usunięcie niektórych cech lub zmiana wymiarowości okazały się negatywnie wpływać na ten model. Należy jednak pamiętać, że klasyfikator ten jest zdecydowanie bardziej złożony od dwóch pozostałych, przez co znalezienie optymalnych hiperparametrów może być w jego przypadku trudniejszym zadaniem. Podczas niniejszej analizy przetestowane zostały jedynie dwa rodzaje jądra - *rbf* oraz *poly* (przy czym dla każdej konfiguracji lepsze rezultaty okazało się dawać *rbf*). Rozsądnym byłoby powtórzenie testów dla większej liczby kombinacji hiperparametrów, szczególnie tych odpowiedzialnych za wymiarowość. Przy odpowiednio dostrojonym modelu wartości metryk mogłyby zbliżyć się do tych osiąganych przez drzewo decyzyjne.

Zgodnie z oczekiwaniami naiwny klasyfikator Bayesa okazał się być najgorszym z badanych klasyfikatorów. Jedynie dla surowych danych osiągnięte wartości metryk pozwalałyby mu konkurować z maszyną wektorów nośnych (która, jak zostało opisane wyżej, była bardzo słabo dostrojona). W przypadku zastosowania selekcji cech lub normalizacji wyniki klasyfikatora znacznie się pogorszyły. Być może zastosowane przekształcenia spowodowały usunięcie istotnych danych lub zaburzyły ich niezależność, na której opiera się idea tego klasyfikatora. Na tym etapie należy jeszcze wspomnieć, że w przypadku naiwnego klasyfikatora Bayesa porównane zostały tylko trzy metody wstępnego przetwarzania danych, a nie pięć, jak w przypadku pozostałych modeli. Wynika to z faktu, że ten klasyfikator nie przyjmuje wartości ujemnych, a te wystąpiły w zbiorach danych poddanych standaryzacji oraz analizie głównych składowych.

6 Wnioski

Dzięki użyciu kilku bibliotek przeznaczonych dla języka Python (*sklearn*, *pandas*, *matplotlib* oraz *seaborn*) jesteśmy w stanie niewielkim nakładem pracy (i kodu) przeprowadzić stosunkowo rozbudowaną analizę danych. Dla niektórych konfiguracji metod przygotowania danych i klasyfikatorów osiągane były bardzo wysokie wartości metryk świadczące o modelach dobrej jakości. W sytuacji, gdzie konieczna byłaby bardzo dokładna analiza danych należałoby sprawdzić, czy model nie uczy się predykować tylko średnie i wysokie zapotrzebowanie na koszulki, ponieważ w ramach tego laboratorium przyjęto podział danych na treningowe i testowe w stosunku 8:2, tzn. dane testowe stanowiły 20% wszystkich danych, a wstępna analiza (tabela 2) wykazała, że koszulki o niskim zapotrzebowaniu to jedynie 11.6% całego zbioru danych. Mogłoby się więc okazać, że akurat wszystkie dane związane z tą kategorią trafiły do części testowej, co negatywnie wpłynęłoby na możliwość obiektywnego predykowania zapotrzebowania na koszulki. Rozwiązaniem tego problemu byłoby odpowiednie użycie parametru `stratify` funkcji `train_test_split` z biblioteki *sklearn.model_selection*. Dodatkowo, podczas próby potencjalnego usprawnienia algorytmu można by w większym stopniu skorzystać z konkretnych własności i zależności danych odkrytych na etapie wstępnej eksploracji. Ostatnim aspektem, który mógłby ulec poprawie byłaby zmiana mapowania poszczególnych kategorii wśród cech koszulek: mapowanie rozmiarów i zapotrzebowania powinno być uporządkowane, np. dla rozmiarów - najmniejszy z nich, czyli 'XS' powinien mieć przypisaną najmniejszą wartość, czyli 0, a następnie wraz ze wzrostem wartości rozmiaru powinny rosnąć także przypisane wartości liczbowe tak, by dla rozmiaru '3XL' przypisana została wartość 6. Podobnie należałoby przekształcić mapowanie dla zapotrzebowania. Jednocześnie konieczna byłaby rezygnacja z tego rodzaju mapowania dla materiałów i kolorów, ponieważ implikuje ono, że większa różnica między wartościami liczbowymi oznacza większy błąd. Zachowanie to jest pożądane w przypadku rozmiarów czy zapotrzebowania, ale zdecydowanie nie jest pożądane dla materiału czy koloru koszulki, gdzie nie można mówić o mniejszym/większym błędzie - jest on dla tych cech zerojedynkowy.